名字嵌入向量方法*

何沧平
(微博, cangping@staff.weibo.com, cphe@lsec.cc.ac.cn)
许涛
(曙光信息产业（北京）有限公司, xutao@sugon.com)

摘　要

在进入推荐系统之前，商品名、人名等实体名字需要嵌入低维向量。word2vec这样的流行嵌入算法的出发点是“相同语法位置上的词具有相似的向量”，而名字序列没有语法结构，导致名字向量的质量不高。本文从“相邻的名字具有相似的向量”出发，提出一个称为*名字嵌入*的新方法。名字嵌入使用了一些新技巧：公式比word2vec更简单，向量模长固定为1、用相对权重处理低频名字、优化目标使用简单的均方差。以名字相似度作为衡量标准，在NBA球队名人造集、球队名微博集和微博点赞集上，名字嵌入均显著优于word2vec。

关键词： 名字嵌入，name2vec，word2vec

Name2vec: Name Embedding for Recommender System

He Cangping
(*WEIBO.COM, cangping@staff.weibo.com, cphe@lsec.cc.ac.cn*)
Xu Tao
(*Dawning Information Industry Co., Ltd, xutao@sugon.com*)

Abstract

Before entering a recommender system, an entity name must be embedded into a vector. Some popular models, such as word2vec, are based on the principle “words which are in the same syntactic position should embedded into similar vectors”. However, sequence of entity names has no syntactic structure, which led to the low quality of name vectors. Based on the principle “neighbouring names should embedded into similar vectors”, this paper proposes a novel algorithm named *name2vec*. Name2vec has some new features: vector length equals 1, relative weight which has solved the low frequency problem, optimization objective function is mean square error rather than cross entropy. The quality of embedding is measured by the similarity entity names. On there datasets from WEIBO.COM, name2vec has a better performance than word2veec.

Keywords: name2vec, name embedding, word2vec

1. 引言与相关工作

当前推荐系统已经广泛使用深度学习技术，商品名、单词、用户名、人名、地名、机构名等名字，都需要先嵌入低维向量，然后才能喂给神经网络。本文将这个嵌入操作称为名字嵌入，将得到的向量称为名字向量。通过计算名字向量之间的相似度，可以向用户推荐商品，推荐同类型的博主，找出内容重复的微博。例如在电商场景下，用户浏览商品“YSL小金条口红”时，为其推

* 2020年10月15日提交.



图 1: 微博的“相关用户”推荐

荐相似的商品“Dior烈艳蓝金唇膏”；在微博场景下，用户搜索博主“来去之间”时，为其推荐一些相关用户(图1)。

常用的名字嵌入方法有几个。矩阵分解算法[1]同时得到用户向量和物品向量；word2vec[2]本来为机器翻译设计，但它也能够得到名词的向量；将word2vec应用于物品集合，并取消训练窗口，就得到了item2vec[3]；Attentive Item2vec[4]在item2vec的基础上添加了注意力。在图神经网络中，节点嵌入低维向量，用节点向量之间的距离反映节间之间的连接关系。图嵌入算法DeepWalk[5]用随机游走生成序列，再用word2vec 的skip-gram模型生成节点向量；LINE[6]和node2vec[7]重新定义了节点相似度。阿里巴巴使用用户浏览的商品序列和元信息为10亿级商品建立向量[8]。

将word2vec类算法应用到实体名字上时，会有一个问题：自然语言句子内部有语法结构，词与词之间的前后顺序不可随意调换。依靠捕捉这个结构，word2vec才能将同一个语法位置上的不同词映射为相似的向量。但是，名字序列中并没有语法结构，名字的先后顺序可以随意调换。例如，“你␣吃饭␣了␣吗”这个自然语言句子中的4个词的前后顺序是固定的，改变顺序后就是病句。而在电商APP上的4个商品名“YSL小金条口红␣带皮腰果␣白腊木餐桌␣五仁月饼”，按任意顺序浏览都不算错。序列中的␣是显式空格，用来分隔词或名字。

自然语言句子中的词语相似，判断依据是它们处在相同的语法位置；而实体相似的判断依据是它们经常相邻出现，如果在用户的浏览序列中，“保温壶”的前后多次出现“保温瓶”，那么可以猜测二者是相似的商品。

本文从“相邻的名字具有相似的向量”出发，设计一个新的名字嵌入算法。该算法采用了一些新的技巧：名字向量的模长均为1，不像word2vec那样允许向量模长大幅变化；每个名字只对应一个向量，不像word2vec那样需要额外的辅助向量；优化目标使用向量夹角余弦的均方差，不像word2vec那样使用向量内积的交叉熵；用名字相对权重来解决低频名字被高频名字带偏问题，不像word2vec那样随机跳过高频单词。在NBA季后赛球队名人造集、球队名微博集上，名字向量间距离与球队对阵关系图符合良好。在微博点赞集上，名字向量间距离与博主共现矩阵相吻合。在这3个序列集上，名字嵌入的表现均超过word2vec.

本文后续内容这样组织。第2节和第3节给出了名字嵌入算法的具体公式，第4节是名字权重的计算方法，第5节给出3个序列集上的实验结果，第6节总结全文。

2. 名字嵌入算法

记名字序列集为S，S包含若干个名字序列，每个序列的长度大于等于2。序列集S对应的字典记为D，即S中出现过的所有名字的集合。字典D中的名字个数记为n。

一个示例序列集S为：{ 雄鹿␣魔术, 雄鹿␣雄鹿, 步行者␣热火␣步行者, 步行者␣热火␣步行者␣雄

鹿, 76人␣凯尔特人␣76人, 快船␣独行侠␣快船␣独行侠␣快船␣独行侠␣快船␣独行侠} 。字典D为：{雄鹿,魔术,步行者,热火,76人,凯尔特人,独行侠,快船}，名字个数$n = 8$.

名字嵌入的目标是将任意名字$u \in D$均映射到一个实数列向量$\boldsymbol{v} \in R^m$，m是任意指定的正整数，约束条件是$\|\boldsymbol{v}\|_2 = 1$，即$\boldsymbol{v}$为单位向量。将字典D中名字对应的n个向量分别记为$\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$。这$n$个列向量从左到右排列，组成一个大小为$m \times n$的矩阵$V$，即$V = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n)$。对任意给定的正整数$k$，将任意给定的$k+2$元组记为$U = (u_1, u_2, \ldots, u_{k+2})$，这里的$u_1$称为正名，$(u_2, u_3, \ldots, u_{k+1})$称为上下文，$u_{k+2}$称为负名。$(u_1, u_2, u_3, \ldots, u_{k+1})$对应名字序列中的一个片段，负词通过负采样算法生成，具体生成方法见第5节。$u_1, u_2, \ldots, u_{k+1}$分别拥有实数权重$w_1, w_2, \ldots, w_{k+1}$，$0 < w_i \leq 1, i = 1, 2, \ldots, k+1$。

当$k = 2$时的一个示例为：$U =$ (步行者, 热火, 雄鹿, 独行侠), w_1, w_2, w_3分别为$0.67, 0.33, 0.67$.

为简化公式，令$\alpha_j = \frac{\min(w_1, w_j)}{\max(w_1, w_j)}, \quad j = 2, 3, \ldots, k+1$.

按照深度学习模型的套路，在$k+2$元组U上设计一个目标函数：

$$h(U) = \frac{c}{k}\sum_{j=2}^{k+1}\alpha_j(\boldsymbol{v}_1^T\boldsymbol{v}_j - 1)^2 + \frac{1}{k}\sum_{j=2}^{k+1}(\boldsymbol{v}_j^T\boldsymbol{v}_{k+2} + 1)^2 + (\boldsymbol{v}_1^T\boldsymbol{v}_{k+2} + 1)^2 \tag{2.1}$$

这里的正实数c为超参数。序列集S上的整体目标函数为

$$H(S) = \sum_{U \in S} h(U). \tag{2.2}$$

求解最优化问题，即可得到名字向量组成的矩阵V^*

$$\begin{aligned} V^* &= \arg\min_{V \in R^{m\times n}} H(S) \\ s.t. & \quad \|\boldsymbol{v}_i\|_2 = 1, i = 1, 2, \ldots, n. \end{aligned} \tag{2.3}$$

U上的损失函数用来观察训练过程中的收敛性，不能包含随机因素，不能直接使用优化目标函数式(2.1)，但又要与式(2.1)保持一致，因此这样定义：

$$l(U) = \sqrt{\sum_{j=2}^{k+1}\alpha_j(\boldsymbol{v}_1^T\boldsymbol{v}_j - 1)^2} = \sqrt{\sum_{j=2}^{k+1}\alpha_j\beta_j^2}, \tag{2.4}$$

这里的$\beta_j = \boldsymbol{v}_1^T\boldsymbol{v}_j - 1$。序列集$S$上的整体损失函数为

$$L(S) = \sum_{U \in S} l(U). \tag{2.5}$$

使用最速下降法等算法求解式(2.3)时，需要目标函数的偏导数，这里一并列出。令

$$\begin{aligned} \beta_j &= \boldsymbol{v}_1^T\boldsymbol{v}_j - 1, \quad j = 2, 3, \ldots, k+1, \\ \gamma_j &= \boldsymbol{v}_j^T\boldsymbol{v}_{k+2} + 1, \quad j = 2, 3, \ldots, k+1, \\ \theta &= \boldsymbol{v}_1^T\boldsymbol{v}_{k+1} + 1, \end{aligned}$$

则有

$$\begin{aligned} \frac{\partial h(U)}{\partial \boldsymbol{v}_1} &= \frac{2c}{k}\sum_{j=2}^{k+1}\alpha_j(\boldsymbol{v}_1^T\boldsymbol{v}_j - 1)\boldsymbol{v}_j + 2(\boldsymbol{v}_1^T\boldsymbol{v}_{k+2} + 1)\boldsymbol{v}_{k+2} \\ &= \frac{2c}{k}\sum_{j=2}^{k+1}\alpha_j\beta_j\boldsymbol{v}_j + 2\theta\boldsymbol{v}_{k+2}. \end{aligned} \tag{2.6}$$

对$\forall i = 2, 3, \ldots, k+1$，

$$\begin{aligned}\frac{\partial h(U)}{\partial \boldsymbol{v}_i} &= \frac{2c}{k}\alpha_i(\boldsymbol{v}_1^T\boldsymbol{v}_i - 1)v_1 + \frac{2}{k}(\boldsymbol{v}_i^T\boldsymbol{v}_{k+2}+1)\boldsymbol{v}_{k+2} \\ &= \frac{2c}{k}\alpha_i\beta_i\boldsymbol{v}_i + \frac{2}{k}\gamma_i\boldsymbol{v}_{k+2}. \end{aligned} \quad (2.7)$$

$$\begin{aligned}\frac{\partial h(U)}{\partial v_{k+2}} &= \frac{2c}{k}\sum_{j=2}^{k+1}\alpha_j(\boldsymbol{v}_j^T\boldsymbol{v}_{k+2}-1)\boldsymbol{v}_j + 2(\boldsymbol{v}_1^T\boldsymbol{v}_{k+2}+1)\boldsymbol{v}_1 \\ &= \frac{2c}{k}\sum_{j=2}^{k+1}\gamma_j\boldsymbol{v}_j + 2\theta\boldsymbol{v}_1. \end{aligned}$$

3. 二元序列上的名字嵌入算法

在有些场景下，序列集S中的每个序列仅包含2个名字，例如微博用户之间的点赞关系（详见见第6节）。此时，训练所用的$k+2$元组退化为3元组，名字嵌入算法公式也有了相对简单的形式，列出如下。

令$\alpha = \frac{\min(w_1,w_2)}{\max(w_1,w_2)}$, 优化目标函数式(2.1)退化为

$$h(U) = c\alpha(\boldsymbol{v}_1^T\boldsymbol{v}_2 - 1) + (\boldsymbol{v}_2^T\boldsymbol{v}_3 + 1) + (\boldsymbol{v}_3^T\boldsymbol{v}_1 + 1)^2. \quad (3.1)$$

令$\beta = \boldsymbol{v}_1^T\boldsymbol{v}_2 - 1$,$\gamma = \boldsymbol{v}_2^T\boldsymbol{v}_3 + 1$, $\theta = \boldsymbol{v}_3^T\boldsymbol{v}_1 + 1$. 偏导数式(2.6-2.8)分别退化为

$$\begin{aligned}\frac{\partial h(U)}{\partial \boldsymbol{v}_1} &= 2c\alpha(v_1^T\boldsymbol{v}_2-1)\boldsymbol{v}_2 + 2(\boldsymbol{v}_3^Tv_1+1)\boldsymbol{v}_3 = 2c\alpha\beta\boldsymbol{v}_2 + 2\theta\boldsymbol{v}_3, \\ \frac{\partial h(U)}{\partial \boldsymbol{v}_2} &= 2c\alpha(v_1^T\boldsymbol{v}_2-1)\boldsymbol{v}_1 + 2(\boldsymbol{v}_2^Tv_3+1)\boldsymbol{v}_3 = 2c\alpha\beta\boldsymbol{v}_1 + 2\gamma\boldsymbol{v}_3, \\ \frac{\partial h(U)}{\partial \boldsymbol{v}_3} &= 2(\boldsymbol{v}_2^T\boldsymbol{v}_3+1)\boldsymbol{v}_3 + 2(\boldsymbol{v}_3^Tv_1+1)\boldsymbol{v}_1 = 2\gamma\boldsymbol{v}_2 + 2\theta\boldsymbol{v}_1. \end{aligned}$$

损失函数式(2.4)退化为

$$l(U) = \sqrt{\alpha(\boldsymbol{v}_1^T\boldsymbol{v}_2-1)^2} = \sqrt{\alpha}|\beta|.$$

4. 名字权重

很多推荐场景中都有长尾现象，小部分名字出现次数高，称为高频名字；大部分名字出现次数低，称为低频名字。高频名字与低频名字相邻时，会有问题。

假设名字u_1、u_2和u_3在序列集S中的出现次数分别为100、2和3，u_1和u_2的相邻次数为1，u_2和u_3的相邻次数为1（见表1）。对u_1来说，只有1%的比例与u_2相邻，那么两个名字的向量$\boldsymbol{v}_1$与$\boldsymbol{v}_2$应该相互远离；对u_2来说，50%的时候与u_1相邻，那么两个名字的向量$\boldsymbol{v}_2$与$\boldsymbol{v}_1$应该相互接近。既远离又接近，矛盾。

对u_2来说，与u_1、u_3相邻的比例均为50%，$\boldsymbol{v}_2$与$\boldsymbol{v}_1$之间的距离应该等于$\boldsymbol{v}_2$与$\boldsymbol{v}_3$之间的距离。根据生活经验，u_2和u_3应该属于同类，u_1属于另一类，$\boldsymbol{v}_2$与$\boldsymbol{v}_3$ 相互接近并远离$\boldsymbol{v}_3$。又一个矛盾。

产生矛盾的根源，是只考虑了相邻名字的一方，忽略了另一方。解决矛盾的办法是同时考虑相邻名字双方的出现次数，赋与它们一个权重。名字权重的定义如下。

对$\forall i = 1, 2, \ldots, n$，名字$u_i$在序列集$S$中的出现次数记为$f_i$。显然$f_i \geq 1$。记$F_1 = \min(f_1, f_2, \ldots, f_n)$。为$u_i$定义台阶数

$$\tau_i = j, \quad 若 2^jF_1 \leq f_i < 2^{j+1}F_1,\ j为正整数.$$

表 1: 名字相邻次数

	u_1	u_2	u_3
u_1	100		
u_2	1	2	
u_3		1	3

表 2: 16个名字的权重，$F_1=4$，$\hat{\tau}=3$

名字u_i	次数f_i	台阶数τ_i	权重w_i	名字u_i	次数f_i	台阶数τ_i	权重w_i
湖人	15	2	0.67	雄鹿	10	2	0.67
开拓者	5	1	0.33	魔术	5	1	0.33
火箭	12	2	0.67	步行者	4	1	0.33
雷霆	7	1	0.33	热火	15	2	0.67
快船	13	2	0.67	猛龙	11	2	0.67
独行侠	6	1	0.33	篮网	4	1	0.33
掘金	19	3	1.00	凯尔特人	17	3	1.00
爵士	7	1	0.33	76人	4	1	0.33

令$\hat{\tau}=\max(\tau_1,\tau_2,\ldots,\tau_n)$，名字$u_i$的权重定义为

$$w_i=\frac{\tau_i}{\hat{\tau}}.$$

作为示例，序列集表3中名字的权重计算过程见表2.

5. 训练过程

1. 预处理序列集S。确保同一个序列中，相邻的名字不相同。每个名字的出现次数f_i均大于等于阈值，阈值为任意指定的超参数。每个序列至少包含2个名字。
2. 对$i=1,2,\ldots,n$，计算w_i，随机初始化$\boldsymbol{v}_i$。指定超参数c，c建议取值范围为[1,10]。指定一个窗口宽度。
3. 在序列集S中的序列上滑动窗口，取出一个序列片断$(u_1,\ldots,u_{\frac{k}{2}},u_{1+\frac{k}{2}},\ldots,u_{k+1})$。对此片断进行负采样 [9]，得到名字$u_{k+2}$，使得$u_{k+2}\neq u_i, i=1,2,\ldots,k+1$。调整名字顺序，即得到$k+2$元组$U=(u_{1+\frac{k}{2}},u_1,,\ldots,u_{\frac{k}{2}},u_{2+\frac{k}{2}},\ldots,u_{k+1},u_{k+2})$。当序列长度为2时，假设序列为$(u_1,u_2)$，那么$k+2$元组为$U=(u_1,u_2,u_3)$，这里的$u_3$通过负采样得到。
4. 以式(2.1)为目标函数，进行1次最优化迭代，更新1次$\boldsymbol{v}_1,\boldsymbol{v}_2,\ldots,\boldsymbol{v}_{k+2}$.
5. 重复步骤3-4，直到损失函数曲线达到满意状态。当损失函数曲线发生震荡时，适当调大c或者调小学习率。

最优化的目标函数形式上是式(2.2)，在得到所有序列片断上的偏导数之后再整体更新所有的$\boldsymbol{v}_i$。实际使用的是式(2.1)，得到每个序列片断上的偏导数之后就更新涉到及的$\boldsymbol{v}_i$，以追求更少的资源消耗和更短的训练时间。

6. 实验

word2vec应用广泛。论文 [3] 已经证明item2vec在物品相似度方面的表现与SVD相当，而item2vec和DeepWalk均是借用word2vec的skip-gram模型，LINE和node2vec的目标函数中也采用了与word2vec相



图 2: NBA季后赛对阵图。来源：新浪体育APP。

表 3: 球队名人造集

序列	重复次数	序列	重复次数	序列	重复次数	序列	重复次数
湖人_开拓者	5	掘金_爵士	7	步行者_热火	4	猛龙_凯尔特人	7
火箭_雷霆	7	掘金_快船	7	雄鹿_热火	5	热火_凯尔特人	6
湖人_火箭	5	湖人_掘金	5	猛龙_篮网	4		
快船_独行侠	6	雄鹿_魔术	5	凯尔特人_76人	4		

同的向量内积。因此，本节选word2vec作为对照基准，训练工具为python模块gensim.models.fasttext。实验所用序列集有3个：球队名人造集、球队名微博集、博主点赞集。

球队名人造集是根据2019-2020 NBA季后赛对阵关系（图2）制作。季后赛是7场4胜制，对阵的2支球队最多打7场，最少打4场。对阵双方的名字组成一个序列，对阵的场数对应序列的重复次数，具体见表3。

名字嵌入算法的配置为：向量维数$m = 2, c = 10$，最速下降法学习率0.001，迭代400轮。word2vec算法的配置为：向量维数2，窗口宽度1，词的最小次数min_count=1，迭代2000轮。所得16个向量的位置关系见图3，图中圆周半径为1。图3(a)中，16个向量大致分为4组：(湖人、开拓者、火箭、雷霆)、(快船、独行侠、掘金、爵士)、(雄鹿、魔术、步行者、热火)、(猛龙、篮网、凯尔特人、76人)，各组内向量的余弦距离较小，各组之间向量的余弦距离较大，与对阵列关系（图2）相吻合。图2显示，半决赛时，湖人对阵掘金、热火对阵凯尔特人；在图3(a)中，这4个名字处于各组的边缘，相互之间距离较近，正确反映了对阵关系。

word2vec生成的16个向量，模长不固定，将它们的模长归一化不影响相互之间的余弦距离。观察图3(b)，独行侠和开拓者之间距离很小，但实际它们并没有对阵关系，与图2不符。还有其它不相符的关系，不一一列举。

球队名人造集中的每个序列都仅包含2个名字，只测试了名字嵌入算法在$k = 1$时的表现。为了测试其在$k \geq 2$的表现，特制作球队名微博集。2019-2020 NBA季后赛的比赛日期为20200816~20200930，在此期间，16支球队的微博官方账号持续发布赛事信息。将这些博主名称、微博文本、人工标签

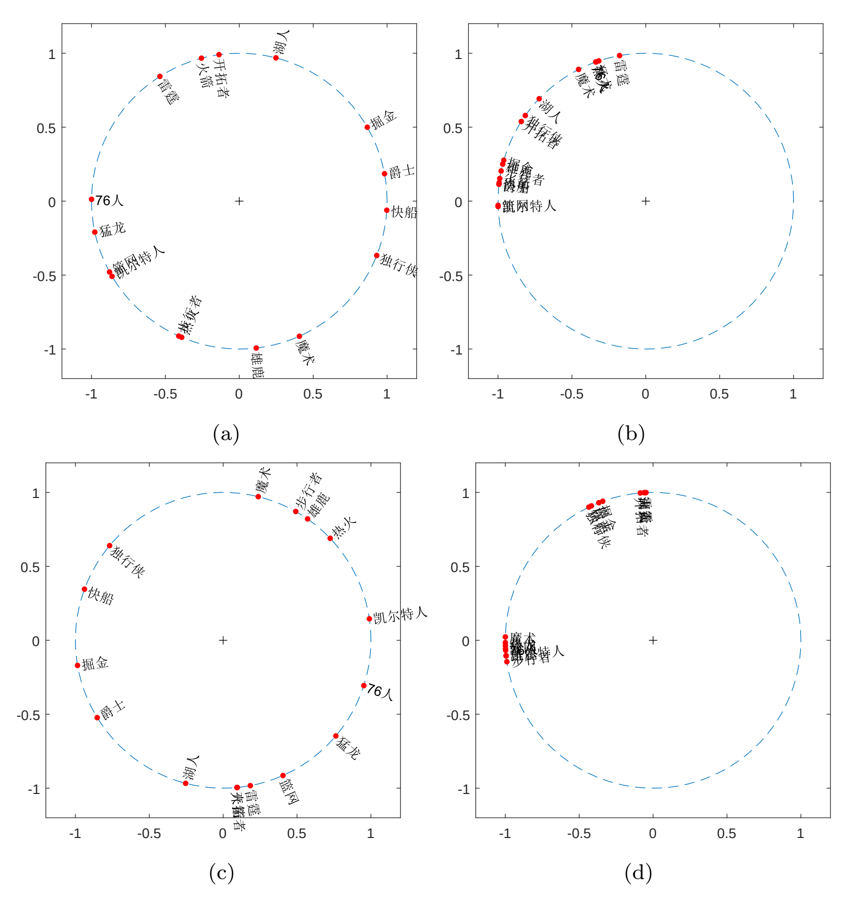


图 3: 球队名对应的向量。(a)对应球队名人造集上的名字嵌入，(b)对应球队名人造集上的word2vec，(c)对应球队名微博集上的名字嵌入，(d)对应球队名微博集上的word2vec。

收集起来，从中提取球队名序列。对任意给定的一条微博，将球队的别名映射为简名，例如，“湖人队”、“洛杉矶湖人”、“洛杉矶湖人队”均映射为“湖人”；有些微博以视频或图片为主，文本中没有球队名，那么采用人工标签中的球队名；如果球队名只包含1个，那么将博主对应的球队名加在序列的头部。共得到序列1706个。包含2个名字的序列有1347个，包含3个名字的序列有202个，包含4至8个名字的序列分别有132个、10个、9个、4个、2个。不同对阵关系的序列数量也不同，见表4，最大数量为270，最小数量为32，相差8.4倍。名字频次最大值为563(热火)，最小值为33(魔术)，相差17倍。

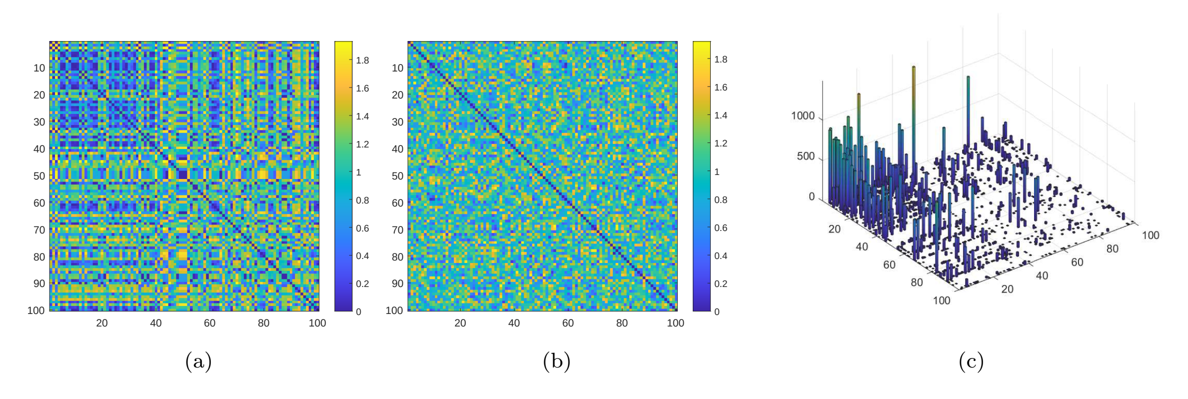
名字嵌入算法的配置为：向量维数$m = 2$, $c = 4$，$k = 2$，最速下降法学习率0.0005，迭代100轮。word2vec算法的配置为：向量维数2，窗口宽度1，词的最小次数min_count=1，迭代1000轮。它们生成的名字向量的分布状态如图3(c-d)。图3(c)与对阵关系相符，图3(d)没有区分出4个组，与对阵关系不符。

博主点赞集中的每一个序列均有2个名字组成，主动点赞的博主名、被点赞的博主名。序列数量1954241，博主名数量60764，博主名频次最大值17476，最小值1，头部集中严重。名字嵌入算法的配置为：向量维数$m = 8, c = 10$，最速下降法学习率0.001，迭代5轮。word2vec算法的配置为：向量维数为8，窗口宽度1，词的最小次数min_count=1，迭代5轮。博主名数量多，无法展示全部向量。这里取前100个博主名的向量，计算它们之间的余弦距离(图4(a-b))，然后与共现矩阵(图4(c))对比，评估向量的优劣。

观察共现矩阵图4(c)，博主名3∼43之间关联密切，博主名3∼43与博主名44∼52联系稀少。图4(a)中，

表 4: 球队名微博集中样本比例

对阵关系	数量	对阵关系	数量	对阵关系	数量	对阵关系	数量
独行侠VS 快船	270	雄鹿VS 热火	164	掘金VS 爵士	74	篮网VS 猛龙	41
快船VS 掘金	219	步行者VS 热火	148	开拓者VS 湖人	61	魔术VS 雄鹿	32
凯尔特人VS 热火	203	湖人VS 火箭	129	湖人VS 掘金	52		
雷霆VS 火箭	170	76人VS 凯尔特人	98	猛龙VS 凯尔特人	45		



(a) (b) (c)

图 4: 博主点赞集上，名字向量之间的余弦距离，(a)对应名字嵌入算法，(b)对应word2vec，(c)是名字的共现矩阵。

博主名3~43之间的距离较小，即较多的蓝色；博主名3~43与博主名44~52之间距离较大，即较多的绿色与黄色，与共现矩阵图相吻合。图4(b)中的向量距离，看起来像随机取值，没有明显的规律性，与共现矩阵不符。

综合本节3个序列集上的实验结果，可以得出结论，名字嵌入优于word2vec.

7. 总结与讨论

与word2vec相比，名字嵌入有几个优点。

更好地体现名字之间的相邻关系，第6节的实验已经说明了这一点。

向量模型均为1，模型压缩时的损失小。fastText [10]对词向量缩方法是，将相近的若干个向量分为一组，例如64个，然后计算这组向量的中心点。使用时，用中心替代这64个向量。这个替代操作会引入误差，在误差绝对值一定的情况下，模长较小的向量的相对误差较大，导致不同向量的压缩误差不同。word2vec产生的向量，模长不固定。在博主点赞集上，word2vec生成的向量的模型最小值为0.007，最大值为6.982，相差1004倍。名字嵌入生成的向量，模长均为1。各个向量的压缩相对误差不会相差太大。

训练资源开销少。word2vec算法中，每个词均对应2个向量，1个词向量和1个隐向量；名字嵌入算法中，每个词只对应1个向量，内存开销少一半。在word2vec算法中，有耗时很多e指数运算或者softmax运算；名字嵌入算法只有简单的加乘运算。

名字嵌入算法的稳定性、收敛速度、高性能的代码实现还有待进一步研究。

参考文献

[1] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems, Computer, 2009

[2] T. Mikolov, K . Chen, G. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. Nips, pp. 1-9, 2013.

[3] O. Barkan and N. Koenigstein. ITEM2VEC: Neural item embedding for collaborative filtering. in IEEE Machine Learning for Signal Processing, MLSP, 2016.

[4] Oren Barkan, Avi Caciularu, Ori Katz, et al. Attentive Item2Vec: Neural Attentive User Representations. arXiv:2002.06205. 2020.4

[5] Bryan Perozzi, Rami Al-Rfou, Steven Skiena. DeepWalk: Online Learning of Social Representations. arXiv:1403.6652. 2014.6

[6] Tang J, Qu M, Wang M, et al. Line: Large-scale information network embedding. arXiv:1503.03578. 2015.3

[7] Aditya Grover, Jure Leskovec. node2vec: Scalable Feature Learning for Networks. arXiv:1607.00653. 2016.7.

[8] Jizhe Wang, Pipei Huang, Huan Zhao, et al. Billion-scale Commodity Embedding for E-commerce Recommendation in Alibaba. arXiv:1803.02349. 2018.3

[9] Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. In Proceedings of NIPS 2013 (pp. 3111-3119)

[10] Armand Joulin, Edouard Grave, Piotr Bojanowski, et al. Bag of Tricks for Efficient Text Classification. arXiv:1607.01759. 2016.7.